

# 采用预测策略的 Earley 算法

谷波 李茹 刘开瑛

(山西大学计算机与信息技术学院 太原 030006)

(计算智能与中文信息处理教育部重点实验室 太原 030006)

**摘要** 在自然语言处理中,句法分析主要有基于统计的方法和基于规则的方法。Earley 算法是一种基于规则的方法,可以分析任意上下文无关文法(CFG),而不需要对文法进行修改。详细分析了 Earley 算法的特点。在通常的 Earley 算法中增加了多种预测机制,这些预测机制借鉴了 LL、LR 以及 SLR 等确定性分析算法的一些思想,并对这几种不同的预测机制及其组合在相同条件下进行了中文句法分析实验。结果显示,引入这些预测机制通常可以减少产生项目的数量,从而节省存储空间,减少运行时间。

**关键词** 上下文无关文法,句法分析,Earley 算法

**中图法分类号** TP391 **文献标识码** A

## Earley Algorithm Using Prediction Strategies

GU Bo LI Ru LIU Kai-ying

(School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China)

(Key Laboratory of Ministry of Education for Computation Intelligence and Chinese Information Processing, Taiyuan 030006, China)

**Abstract** There are two kinds of parsing algorithms in nature language processing: one based on statistics and the other based on grammar rules. Earley algorithm is based on grammar rules. It can parse any context free grammar (CFG) without changing the grammar. This paper used several predictive strategies in Earley algorithm. These strategies come from LL, SLR and LR algorithms. Experiments were made for these strategies and their combinations. Results indicate that these strategies can usually reduce number of items and make parsing faster.

**Keywords** Context free grammar, Parsing, Earley algorithm

## 1 引言

句法分析是自然语言处理领域中很重要的一个环节。它前承分词、词性标注,后接语义分析、篇章分析和信息抽取等。在中文信息处理中,中文分词和词性标注技术已经趋于成熟,语义分析和篇章分析也有许多学者在进行研究。但是中文句法分析难度较大且目前还不很成熟,因此已经成为制约语义分析和篇章分析等后续信息处理的瓶颈。句法分析的方法可以分为两类:一类是基于统计的方法<sup>[1]</sup>,另一类是基于规则的方法。目前有许多新的统计机器学习方法应用于句法分析,如 SVM<sup>[2]</sup>,ME<sup>[3]</sup>,MEMM<sup>[4]</sup>等。单纯使用统计方法难以将句子处理成嵌套形式的短语结构,因此这些方法通常进行的是浅层句法分析。基于规则的方法可以很方便地将短语组织成嵌套形式,从而实现完全句法分析。对于基于规则的方法而言,存在的困难主要有以下两个方面的因素。

第一,中文句法分析可使用的短语语料库比较匮乏。中文句法规则可以由语言专家人工给出,也可以从人工标记短语的语料库中进行自动抽取。前者实现起来比较困难,很难覆盖短语的组成情况,也很难获得其他一些有用的统计信息。

后者实现相对容易,当语料库规模较大时,可以很好地覆盖实际中的短语构成情况,并且能够从中获取相关统计信息。但短语语料库除了分词和标注词性之外还要进行多层嵌套的短语标注,这对于标注人员来说是一件费时费力的事情。目前国内很难获得一个规模比较大的、可用的中文短语语料库。

第二,句法分析算法本身效率不高。许多高效的确定性算法都是针对上下文无关文法(缩写为 CFG)的一些特定子集的,通常自然语言无法用这些子集来描述。确定性算法在其特定的 CFG 子集上的时间复杂度可以达到  $O(n)$ ,其中  $n$  为输入字符串中字符的个数,如 LR 算法<sup>[5]</sup>等。在自然语言处理中使用的是一些针对 CFG 的通用句法分析算法,主要有 CKY 算法<sup>[6]</sup>、GLR 算法<sup>[7]</sup>和 Earley 算法<sup>[8]</sup>等。这些算法的特点是能处理任意的 CFG,但其时间复杂度较高,一般情况下为  $O(n^3)$ 。CKY 算法可从句子的任意位置开始进行分析,但要求文法必须是乔姆斯基范式(CNF)。虽然可以将任意一个 CFG 改为 CNF,但在自然语言中这样做有可能改变整个短语类型标记体系和原有的短语结构。GLR 算法是一种改进的 LR 算法,当 LR 分析表中出现冲突时,建立多条并行分析路径使得分析继续进行。GLR 算法使用图栈和共享节点

到稿日期:2009-02-25 返修日期:2009-05-06 本文受国家 863 计划(2006AA01Z142),国家自然科学基金项目(60873128)资助。

谷波(1978-),男,博士生,讲师,主要研究方向为自然语言处理等,E-mail:gubo@sxu.edu.cn;李茹(1963-),女,教授,主要研究方向为人工智能、自然语言处理等;刘开瑛(1931-),男,教授,博士生导师,主要研究方向为自然语言处理等。



时对起始项目进行限制,称这种策略为 LL 策略。预先求出文法每个非终结符的 First 集,当进行 Predictor 操作时,检查非终结符所导出的每个规则,根据不同的情况来判断是否在当前状态中添加这个规则产生的起始项目。具体方法是:假设从输入句子中读到的当前终结符是  $x$ ,如果规则的右部第一个是终结符且是  $x$ ,或者规则的右部第一个是非终结符且该终结符的 First 集中含有  $x$ ,则对此规则产生新的起始项目并加入到当前状态中,否则不进行。

第二种策略,采用 Follow 集对 Completer 操作进行归约上的限制,称为 F 策略。事先求出每个非终结符的 Follow 集。当对当前状态中的归约项目进行 Completer 时,假设当前归约项目形如  $[A \ X_1 \dots X_m \ , i]$ ,且当前读入的终结符为  $x$ ,若  $A$  的 Follow 集中含有  $x$ ,则对这个归约项目进行 Completer 操作,否则不进行。

第三种策略,采用展望符集对 Completer 操作进行归约上的限制,称为 L 策略。虽然展望符集的计算相对 Follow 集而言更为复杂,但展望符集更为精确地指出了归约之后非终结符后面可能出现的终结符。原始的 Earley 算法中,从每个起始项目开始就计算其展望符集,移过非终结符或终结符时,将展望符集传给后面的项目;遇到归约项目时,根据归约项目的展望符集中是否有当前输入的终结符决定是否对其实行 Completer。通过分析对展望符集的计算过程做了如下一些改进。

非归约项目的展望符集对其自身并没有作用,它们只是起到向后面的项目传递展望符集的作用。这些展望符集一方面会浪费存储空间,另一方面每次的传递也会增加时间上的开销。所以并不是从一开始就计算每个起始项目的展望符集,而是将展望符集的计算推迟到遇到形如  $[A \ X_1 \dots X_m \ , i]$  的项目时才进行,称这种项目为“次归约项目”。这样,当进行归约时,通过归约项目,左指针查找其对应的次归约项目即可获得展望符集。之所以求次归约项目的展望符而不是直接去求归约项目的展望符,是因为如果遇到归约项目时再去计算展望符集,可能导致展望符集不能直接求得。例如,当遇到项目  $t = [A \ X_1 \dots X_m \ , j]$  时计算展望符集,那么要考虑  $S_j$  状态中所有后面是  $A$  的项目。把这些项目分为两类:一类形如项目  $s = [B \ \dots AC \dots , i]$ ,另一类形如  $r = [B \ \dots A \ , i]$ 。因为自然语言文法通常不含空导符号,所以对于前一类,如果  $C$  是终结符,就将  $C$  加入到项目  $t$  的展望符集;如果  $C$  是非终结符,就将  $\text{First}(C)$  并入项目  $t$  的展望符集。对于后一类情况,应该将  $r$  的展望符集并入  $t$  的展望符集,但此时  $r$  的展望符集没有事先计算出来,所以这时候无法继续去求  $t$  的展望符集,这说明次归约项目  $r$  的展望符集应该提前求出。因此,将展望符集的计算推迟到出现次归约项目时再求次归约项目的展望符集,就不存在上述求不到的问题。若次归约项目有了展望符集,那么归约项目就可以根据 left 指针直接找到次归约项目,获得其展望符集。下面是改进后的求展望符集的算法。

1)对于初始项目  $[S \ S, 0]$ ,其展望符集为  $\{ \$ \}$ ,  $\$$  表示句子末尾;

2)对于新产生的次归约项目  $t = [A \ X_1 \dots C \ , j] \ S_i$ ,按如下方式计算其展望符集:

a)若  $[B \ X_1 \dots AX_m \dots , k] \ S_j$ ,且  $X_m \in VN$ ,则将  $X_m$  的 First 集并入  $t$  的展望符集;若  $X_m \in VT$ ,则将  $X_m$  添加到  $t$  的展望符集;

b)若项目  $s = [B \ X_1 \dots A \ , k] \ S_j$ ,则将  $s$  的展望符集并入  $t$  的展望符集。

上面第一种策略修改了 Predictor 操作,第二种和第三种策略需要修改 Completer 操作。改进后的操作如下。

1)Predictor:设  $[A \ X_1 \dots C \dots X_m \ , j] \ S_i, C \in VN$ ,当前读入的终结符为  $x$ 。对文法中每个形如  $C \ Y_1 \dots Y_n$  的规则,若  $Y_1$  是终结符  $x$ ,或  $Y_1 \in VN$  且  $Y_1$  的 First 集中含有  $x$ ,则在状态  $S_i$  中增加新的形如  $[C \ Y_1 \dots Y_n \ , i]$  的项目;

2)Completer:设  $[A \ X_1 \dots X_m \ , j] \ S_i$ ,当前读入的终结符为  $x$ 。若  $A$  的 Follow 集(策略二)或该项目左指针指向的次归约项目的展望符集(策略三)含有  $x$ ,且  $[B \ X_1 \dots A \dots X_m \ , k] \ S_j$ ,则在  $S_i$  中增加  $[B \ X_1 \dots A \dots X_m \ , k]$ 。

每种策略都可以单独使用。第一种策略可以结合第二种或者第三种策略,第二种和第三种策略不能同时使用。另外需要说明的是,LL 策略仅需要求 First 集,求 Follow 集时需要用到 First 集,这两个集合可以根据文法直接求得而待分析的句子无关。展望符集的计算也需要用到 First 集,而且需要在分析句子的时候根据状态的变化逐步求出,不能事先求得。我们在实验中计算算法运行时间的时候,将其中各种集合的计算时间都包含在内。

## 5 实验结果及分析

在两个不同的中文短语文法上进行了实验。第一个文法来自中科院计算所,共有 376 条规则。其中的规则比较长,最长的规则右部有 7 个文法符号。在此感谢中科院计算所提供的文法。另一个文法是我们自己写的一个文法,其中共有 169 条规则,规则右部长度不超过 3 个文法符号。使用这两个文法在实际分析句子时,大多数句子都会产生多个分析结果。通常句子越长,分析结果越多,导致算法运行时间特别长。为了减少分析时间,没有在长的句子上进行实验,仅在词数为 2 个至 6 个的短句或短语上进行了实验,每个词数下的句子数不尽相同。所有算法都在相同的硬件和软件环境下运行,并且算法均采用 Perl 实现。表 1 列出了每个文法上不同词数下的句子个数。

表 1 实验句子数

句数	2 词	3 词	4 词	5 词	6 词
文法 1	127	48	36	10	5
文法 2	168	100	80	39	34

为了检验这几种预测机制对算法的影响,分别测试了采用 LL 策略(LL E),LL + F(LLFE),LL + L(LLLE),Follow (FE),L(L E)的 Earley 算法以及没有采用任何预测策略的 Earley 算法(E)。实验结果平均到每个句子。

图 1 和图 2 分别是两个文法集上不同词数下平均每个句子产生的项目数。由于项目占据了主要的存储空间,项目数可以反映出算法在空间上的效率。

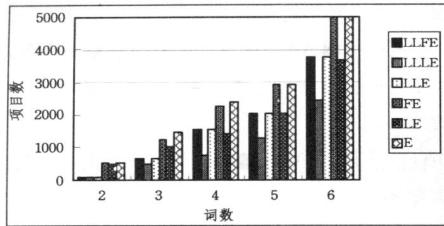


图1 文法1产生的项目个数

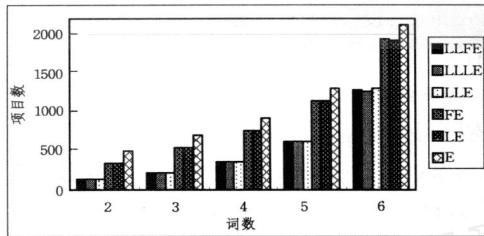


图2 文法2产生的项目个数

表2和表3是两个文法集上不同词数下平均分析每个句子所花费的时间(单位是秒),行表示不同词数,列表示不同算法。运行时间反映出了算法的时间效率。

表2 文法1花费的时间

秒	LLFE	LLLLE	LLE	FE	LE	E
2	0.11	0.11	0.12	0.73	1.31	1.53
3	0.81	0.85	0.81	3.88	3.69	6.67
4	3.50	2.22	5.14	9.43	7.72	35.92
5	3.60	3.30	6.40	41.60	15.40	41.60
6	5.00	4.00	8.00	67.00	24.00	67.00

表3 文法2花费的时间

秒	LLFE	LLLLE	LLE	FE	LE	E
2	0.05	0.06	0.05	0.29	0.31	0.50
3	0.09	0.11	0.09	0.46	0.52	0.70
4	0.14	0.20	0.14	0.70	0.80	1.10
5	0.28	0.44	0.28	1.28	1.31	2.10
6	1.59	1.78	0.97	2.76	3.26	7.51

先从算法产生的项目数量上看,图1显示出LFE与没有使用预测策略的Earley算法(E)基本差不多,LLLLE结果最好,其它3种基本差不多,但都要明显优于E;图2显示出LLE,LLLLE和LLFE这3种算法效果都比较好且性能相近,FE和LE效果不如前3种,但比E要稍好些。3种策略中,LL策略自顶向下限制无用起始项目的出现,使L策略和F策略可以自底向上限制对无用归约项目的规约。因为展望符集比Follow集精确,所以相同条件下使用L策略去掉的项目数肯定大于等于使用F策略去掉的项目数。一般情况下LL策略和L策略(或F策略)去掉的冗余项目不会完全一样,所以就“减少项目数”而言,LLLLE是不会比其它的算法差的。从实验结果来看,LLLLE产生的项目数平均为E的一半。

再从时间上看,表2的结果和图1的结果反映出了时间和项目数上的一致性,即产生的项目数量越少,算法花费的时间也越少;表3总体上反映了这一情况,但当LLE,LLLLE

和LLFE在产生项目数量相差不多时,LLE花费的时间最少,而LLLLE花费时间最多。这说明对文法2来说,大部分项目已经由LL策略过滤,而F策略和L策略几乎不起作用,这时对归约项目进行限制等于花费了额外不必要的时间。从实验结果来看,表2中LLLLE比E快13倍左右,而表3中LLE比E平均快8倍。

LL策略是一种实现简单且效果较好的策略,虽然从理论上说即使不使用该策略,那些可以被其过滤的无用起始项目也会在Scanner时被过滤掉,但是后面状态进行Completer时会花费时间来扫描这些项目。如果这些无用起始项目数量众多,那么扫描花费的时间自然也就很多。自然语言的文法中非终结符数量不多,但其规则数目通常比较多,表明每个非终结符对应的规则特别多。而LL策略能够极大地限制某些规则对应的无用起始项目的出现。展望符集比Follow集更精确,使用L策略过滤掉的项目数不会少于使用F策略过滤掉的项目数。限制Completer只会限制当前的一个归约项目,而限制Predictor通常会限制一批由当前非终结符所导出的项目。这就解释了LLE在总体上要优于LE或FE。

结束语 这些算法在时间和空间的效率表现上基本一致,但当某些算法产生的项目数相近时,算法在时空上的效率会有所差别。一方面说明项目数的减少意味着算法的查找时间会减少,另一方面说明对一些特定的文法或句子,不同算法产生的项目数相差无几时,策略本身所花费的时间开销会显现出来。实验结果显示,LL+L策略在减少项目个数上是最好的,虽然单纯使用LL策略在时间效率上比较高,但综合考虑而言LL+L策略效果通常较好。

## 参考文献

- [1] 孟遥,李生,赵铁军,等.基于统计的句法分析综述[J].计算机科学,2003,30(9):54-58
- [2] 李珩,朱靖波,姚天顺.基于SVM的中文组块分析[J].中文信息学报,2004,18(2):1-7
- [3] 李素建,刘群,杨志峰.基于最大熵模型的组块分析[J].计算机学报,2003,26(12):1722-1727
- [4] Sun Guang-lu, Huang Chang-ning, Wang Xiao-long, et al. Chinese Chunking Based on Maximum Entropy Markov Models [J]. Computational Linguistics and Chinese Language Processing, 2006, 11(2): 115-136
- [5] Aho A V, Johnson S C. LR Parsing [J]. Computing Survey, 1974, 6(2): 99-124
- [6] Graham A L, Harrison M A, Ruzzo W L. An improved context-free recognizer[J]. ACM T PROGR LANG SYS, 1980: 415-462
- [7] Tomita M. An Efficient Augmented-Context-free Parsing Algorithm[J]. Computational Linguistics, 1987, 13(1): 31-46
- [8] Earley J. An efficient context-free parsing algorithm[J]. Communications ACM, 1970, 13(2): 94-102
- [9] Aho A V, Lam M S, Sathi R, et al. Compilers Principles Techniques and Tools(2nd) [M]. New York: Addison-Wesley, 1986: 191-300